A Method to Suggest Alternative Routes Based on Analysis of Automobiles' Trajectories

João Pedro Schmitt
Santa Catarina State University
Graduate Program in Applied Computing
Joinville, Brazil
schmittjoaopedro@gmail.com

Fabiano Baldo
Santa Catarina State University
Graduate Program in Applied Computing
Joinville, Brazil
fabiano.baldo@udesc.br

***Abstract*—Inexperienced drivers usually use the most-known paths to move inside the cities, while drivers with a better knowledge of the road network normally taken alternative routes that are shorter, faster or safer. This knowledge about roads usage, when shared with other drivers, could offer more paths options to distribute the traffic load across the city by suggesting alternative routes. However, the problem lies in how to suggest alternative route directions for ordinary drivers considering knowledge gathered from experienced drivers. In order to try to solve this problem, it is proposed an algorithm, named TODS - *Trajectory Outlier Detection and Segmentation*, to group and segment car road trajectories in standard and alternative routes based on city roads usage in different day times periods. After that, the segmentation results are suggested as driving directions for ordinary drivers. To evaluate the results was performed a qualitative comparison with TRA-SOD algorithm considering the segmentation process. The tests were executed using two trajectories datasets collected by drivers in San Francisco - USA and Joinville - Brazil. The results assessment indicate that TODS is superior to TRA-SOD due to its segmentation characteristics. Besides that, it has been observed that the time period of the day influences how routes are used along the day.**

***Index Terms*—Trajectory similarity, moving objects analysis, frequency-based clustering, standard segments, alternative segments, real-world scenarios**

I. Introduction

The study of road traffic planning focuses on understanding how people use the city paths to make improvements and arrangements in the road network [25]. However, this task is becoming even more difficult with the increasing number of vehicles in the cities. But, with the growth of big data field due to investments in software, hardware and services, combined with the advance of data mining and data analysis techniques, new opportunities to improve people's lives are appearing [22]. Due to this fact, the use of data analysis techniques is getting even more important in the study of road traffic users behavior.

In the traffic planning field is necessary to know when, where and how people travel daily inside the city in order to reorganize the distribution of the traffic flow. Investigating the traffic behavior, a possible factor that causes traffic jams could be the intensive use of main roads by ordinary drivers that do not know alternative routes. Usually, these drivers make use of navigation devices to suggest routes with shortest paths or, more recently, with congestion constraints. But, there are situations where the paths suggested by these devices hide the existence of less used paths, that could improve the traffic distribution flow when suggested as alternative routes.

In contrast to ordinary drivers, there are the expert drivers that usually know pretty well, partially or entirely, the city road network. These expert drivers are seen as outliers due to their ability to get alternative routes, depending on the time of the day or the day of the week, that lead to shorter, faster and even safer roads. So, the use of this knowledge could be applied to decrease congestions as well as to improve mobility inside the cities, mitigating or avoiding the necessity of road network arrangements by public traffic offices. Besides that, this knowledge could improve the answers of navigation systems, as Google Maps [6], providing suggestions of near less frequently used alternative trajectories from the standard route directions, like shortcuts and deviations.

Based on the aforementioned assumptions, this work addresses the following research question: how to suggest alternative routes for ordinary drivers considering knowledge gathered from trajectories of experienced drivers? To answer this question clustering techniques from the data mining field are used to approach this problem. Clustering is an unsupervised technique widely adopted in scenarios where it is necessary to find patterns or outliers in datasets. Therefore, this work proposes an algorithm to group and segment car road trajectories applying a clustering technique based on the collected data. After that, the clusterization results are used to suggest standard and alternative routes as driving directions for ordinary drivers. This algorithm is named as Trajectory Outlier Detection and Segmentation (TODS).

Actually, one of the main problems to separate trajectories segments in standard and alternative relies on how to calculate and compare the similarity between them. Several studies propose to analyze alternative trajectories [2], [4], [7], [15], [16], but none of them evaluate the alternative trajectories segmentation based on the road usage level. To tackle this problem, an approach for grouping users' trajectories by distance is performed in order to identify standard paths with high vehicles flow and alternative segments with fewer vehicles flow.

The algorithm proposed here is evaluated using automobiles trajectories collected in San Francisco - EUA and Joinville - Brazil. A qualitative evaluation is executed to analyze the

TODS segmentation process considering the temporal evolution (along time periods of the day), while a quantitative analysis evaluates run-time statistics. In addition, the TODS is compared to one of the most know benchmarks in the literature, the TRA-SOD algorithm [4].

The next sections are organized as follows. Section II presents the related works. Section III define the terms and notations of this study. Section IV presents the proposed algorithm, and section V presents the performed tests and results. Finally, section VI outlines the conclusion and future works.

II. Related works

Various studies have already addressed the identification of outliers in trajectory analysis. A summary of these studies is presented bellow.

Ma et al. [11] propose the use of Local Outlier Factor (LOF) density technique combined with Principal Component Analysis (PCA) to find errors and anomalies analyzing images from automotive traffic data captured by cameras. The proposed method presented a 93.5% hit rate. Fontes et al. [4] propose an algorithm (TRA-SOD) that uses the space and space-time semantic analysis to find outliers between trajectories moving together between regions on the map. This technique classifies as default the trajectories that have a minimum amount of neighbors grouped in a cluster. In contrast, trajectories that do not meet this minimum number of neighbors are classified as outliers. Qualitative analyses indicate that several outliers were classified as shortcuts and stoppages routes. Zhu et al. [16] propose an online algorithm (TPRRO) to detect abnormal trajectories using historical data and popular routes. This method focuses on space and time anomalies. The time semantics context allows to find outliers for considering different periods of the day that could be unidentified without the use of time context. The authors also suggest that the algorithm can be improved by adding the sub-trajectory (segments) processing. Bessa et al. [2] use convolutional neural networks to detect outliers in bus lines. The study proposes a visual tool to help users to visualizes bus drivers anomalous behaviors. However, due to the use of supervised techniques some data mining operation can be negatively affected by lack of road network information from trajectories. Lee et al. [7] propose a framework based on partitioning and detection to find outliers. This framework divides trajectories in segments sets and classifies the outliers based on the similarity between them. The qualitative results obtained by visual inspection indicate that the addition of the time context can improve the classification of outliers. Finally, Yuan et al. [25] propose a method to find the best route between an origin and destination point based on the analysis of taxi drivers route preferences. The method suggests the best driving direction considering a graph where the vertices are city landmarks and the edges are routes that connect them. The routes are calculated according to the road frequency of use, total travel time, weather conditions, context of the day (weekday or weekend) and the time of the day. The method presents 60 to 70 percent better suggestions when compared with other methods from literature.

In contrast with the related works, the presented study uses a different approach to find alternative (outliers) directions. The proposed algorithm, named TODS, evaluates trajectories path utilization using clustering and segmentation techniques based on trajectories grouping by usage frequency in different time intervals. So, it is possible to evaluate the standard roads utilization in different periods of the day. Besides that, the trajectories segmentation distinguishes between standard and alternative paths, this segmentation technique allows users to identify alternative trajectories segments that could be used as shortcuts and deviations by drivers.

III. Definitions

This section describes the definitions applied in the TODS algorithm.

Definition 1 - Point: A point p is a tuple $p_i = (x_i, y_i, t_i)$, where x and y are geospatial coordinates and t is the time (year, month, day, hour, minute e second) when the point was collected [4].

Definition 2 - Trajectory: A trajectory T is given by $n > 1$ points $\left[p_1, p_2, \ldots, p_n\right]$ collected over time, where $t_1 < t_2 < \ldots < t_n$ [4]. Besides that, for a given trajectory T_j its start time is given by $TS_{T_j} = t_1$ and its end time is given by $TE_{T_j} = t_n$.

Definition 3 - Segment: A segment S in a trajectory T is a list of consecutive points $\left[p_k, p_{k+1}, p_{k+2}, \ldots, p_m\right]$ where $\forall i, k \leq i \leq m, p_i \in T$ [4].

Definition 4 - Region: A region RG is a rectangular area composed by $\left[p_i, p_j\right]$, where p_i and p_j points are the rectangle lower left corner and the top right corner, respectively.

Definition 5 - Route Candidate: A route candidate C is a segment $S = \left[p_k, p_{k+1}, \ldots, p_m\right]$ from one trajectory T that crosses two given regions RG_{start} and RG_{end}, where $RG_{start} \cap RG_{end} = \emptyset$, within a given time interval defined by δ_{start} and δ_{end} with $\delta_{start} < \delta_{end}$. To compute the candidate minimum amount of trajectory points linking two given regions respecting the time interval, the time difference between the first and the last candidate points are calculated by (1) in order to get the last point inside the RG_{start} and the first point inside the region RG_{end}.

$$\begin{aligned} &\underset{k,m}{\text{minimize}}\ \text{timeDiff}(k,m) = t_m - t_k \\ &\text{subject to } k < m \\ &\qquad \wedge p_k \subseteq RG_{start} \wedge p_m \subseteq RG_{end} \\ &\qquad \wedge t_k > \delta_{start} \wedge t_m < \delta_{end}, \\ &\qquad (k,m) \in C. \end{aligned} \tag{1}$$

Definition 6 - Group: A group G is a set of candidates $G = \left[C_1, C_2, \ldots, C_l\right]$, with $l \leq N$ (N is the number of candidates), where $\forall p_i \in C_k \wedge C_k \in G$ exists another point $p_j \in G \wedge p_j \notin C_k$ with $dist(p_i, p_j) < D$ [8], [9], [12], [15]. In this work it has been use as distance function the Euclidean distance (2):

$$dist(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

The group size SZ_G is defined as the number of candidates $C \in G$.

Figure 1 shows a grouping example. In this case the group G_1 contains the candidate $[C_1]$ and the group G_2 contains the candidates $[C_2, C_3, C_4]$. Therefore, the group size of both are $SZ_{G_1} = 1$ and $SZ_{G_2} = 3$.

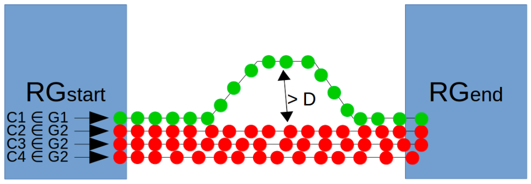


Fig. 1. Candidates grouping

Definition 7 - Standard and alternative Group Sets: Standard and alternative group sets, named STG and $NSTG$, respectively, are defined as follows. Given the whole group set $GT = [G_1, G_2, G_3, \ldots, G_h]$ where $h \leq N$ (N is the number of candidates), ordered according to $SZ_{G_1} \geq SZ_{G_2} \geq \ldots \geq SZ_{G_h}$, the STG is determined by $STG = [G_1, \ldots, G_k]$ with a user defined number of k standard groups, where $k \leq h$, and $NSTG$ is determined by $NSTG = [G_{k+1}, G_{k+2}, \ldots, G_h]$. So, the set of standard groups is composed of the groups with most candidates number while the other groups compose the alternative set.

Definition 8 - Route: A route R_i represents a path that connects a start region RG_{start} and an end region RG_{end} to be used as driving direction for an ordinary driver. The routes are classified as standard and alternative routes. A standard route R_i is derived from every candidate $C_i \in STG$, while an alternative route R_i is derived from every candidate $C_i \in NSTG$. Alternative routes may have part of their trajectories in common with standard routes. Therefore, it is necessary to determine for the candidates that belong to the $NSTG$ which segments make intersection with candidates from STG in order to classify them as standard or alternative segments. This is a two-step approach performed based on literature available in [1], [3], [13], [14], [18]. In the first step, an alternative route R_i is segmented by distance resulting in a set of standard segments $DS_{R_i} = [S_\alpha, S_{\alpha+1}, \ldots, S_{\alpha+l}]$, where $\forall p_i \in DS_{R_i}, \exists p_k \in STG$, $dist(p_i, p_k) < D$, and a set of alternative segments $NDS_{R_i} = [S_\beta, S_{\beta+1}, \ldots, S_{\beta+q}]$, where $\forall p_i \in NDS_{R_i} \wedge \forall p_k \in STG$, $dist(p_i, p_k) > D$. Figure 2 exemplifies the segmentation of C_1 by distance that results in the route R_1. In this case, the route R_1 is segmented in two sets, the set of standard segments $DS_{R_1} = [S_1]$ and the set of alternative segments $NDS_{R_1} = [S_2]$. Besides that, Figure 2 also presents the standard route R_2 derived from the standard candidate C_2. As C_2 belongs to the STG it does not have alternative segments.

In the second-step, the angular distance is used to improve the route segmentation by re-calculating the alternative segments' starts and ends. This is necessary because Euclidean distance could present shortcomings. Figure 2 exemplifies a wrong segmentation of route R_1 using distance. As can be seen, it is missed the real moment where the alternative trajectory diverges from the standard group. Works like [3], [14], [18] present approaches to segment trajectories having into account angular distance. In cases like those presented in Figure 2, the solution is to go backward in C_1 until to find the point where it diverges from the standard path C_2, comparing the angular difference between alternative and standard segments using (3).

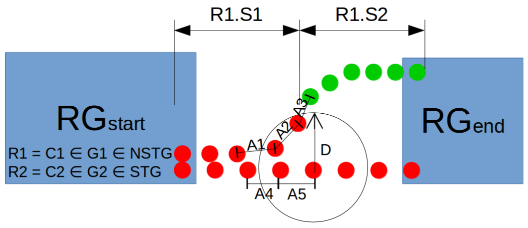


Fig. 2. Candidate segmentation by distance

$$atan2(x,y) = \begin{cases} arctan(\frac{y}{x}), & \text{if } x > 0 \\ arctan(\frac{y}{x}) + \pi, & \text{if } x < 0 \text{ and } y \geq 0 \\ arctan(\frac{y}{x}) - \pi, & \text{if } x < 0 \text{ and } y < 0 \\ +\frac{\pi}{2}, & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2}, & \text{if } x = 0 \text{ and } y < 0 \\ undefined, & \text{if } x = 0 \text{ and } y = 0 \end{cases} \tag{3}$$

Figure 3 exemplifies the application of angular difference to improve the segmentation process. In this example it is compared the angular difference of the alternative edge A_2 and the nearest standard edge A_5. As the angular difference of them is greater than a threshold θ, the points of A_2 are added to the alternative segment. However, the angular difference between edge A_1 and A_4 is less than θ, so the points from A_1 are kept in the standard group set.

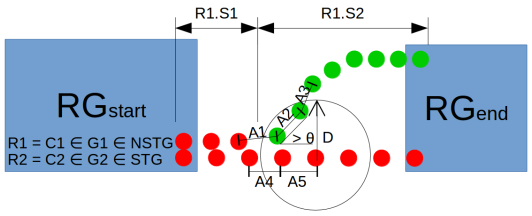


Fig. 3. Candidate segmentation by angular difference

IV. TODS - Trajectory Outlier Detection and Segmentation Algorithm

This study proposes an algorithm, named TODS, to detect and segment alternative trajectories between two regions, considering time interval context. The main algorithm is presented in Algorithm 1 and its inputs are defined as follows:

- Start region (SR) and end region (ER): represent the rectangular regions where the selected trajectories should cross (Definition 4);
- Start hour (SH) and end hour (EH): represent the time when the trajectories should cross SR and ER, they must respect the inequality $SH < EH$;
- Interpolation size (I), standard deviation (SD) and sigma (σ): are used to apply filters and pre-process the trajectories;
- Clustering distance (D), angular difference threshold (θ) and standard groups number (KS): are used to identify the segments of alternative trajectories.

The algorithm output is two sorted sets of standard and alternative trajectories, additionally with their respectives segments classified accordingly. The next sections presents more details about each algorithm phase.

Algorithm 1 TODS - Algorithm

```
Require: $SH, EH, SR, ER, I, SD, \sigma, D, \theta, KS$
 1: $t \leftarrow FindTrajectories(SR, ER)$
 2: for $i \leftarrow 0$ to $Length(t)$ do
 3:     $FilterNoisePoints(t[i], SD, \sigma)$
 4:     $InterpolatePoints(t[i], I)$
 5: end for
 6: $C \leftarrow GetCandidates(t, SH, EH, SR, ER)$
 7: $idx \leftarrow CreateClusteringGrid(C, D)$
 8: $GT \leftarrow GetTrajectoriesGroups(C, idx)$
 9: $SGT \leftarrow GetStandardTrajectories(GT, KS)$
10: $R \leftarrow GetTrajectoriesRoutes(GT, SGT, D, \theta)$
11: return $SGT, R$
```

A. Recovering

The recovering phase is executed by lines range 1-7 presented in Algorithm 1. This phase starts obtaining raw trajectory data from an external source ($FindTrajectories$ line 1). In this study, the external database system used was the MongoDB [17], due to their capacity to execute geometrical queries to retrieve all stored trajectories that intersect two given distinct rectangular geometries (SR and ER).

After that, the method $FilterNoisePoints$ (line 3) is used to remove invalid points or even the entire trajectory in a two steps approach. The first step calculates a normal distribution using the distance between all adjacent points (using Equation 2). The points that extrapolate the confidence interval defined by σ are removed. The second step calculates the trajectory points standard deviation to remove completely trajectories that overpass the defined SD parameter.

The $InterpolatePoints$ function (line 4) is applied after $FilterNoisePoints$ to normalize the distribution of points per space and time for each filtered trajectory. The I parameter defines the upper bound distance between two adjacent points. Thus, if the distance between two adjacent points is greater than the value of I, an interpolation is executed to insert as many middle points as necessary to fit the distance of point shorter than I. The interpolation increases the trajectories points density in order to improve the classification performed by the clustering algorithm in the next step. It is suggested to use an I value lower than the distance value, parameter D, to guarantee that two distinct trajectories with intercalated points will be classified by the clustering algorithm.

The next step is the $GetCandidates$ function (line 6), that removes trajectories that do not meet the time interval given by SH and EH. Besides that, this function also filters segments that respect the trajectory direction, leaving SR and going to ER.

Finally, the $CreateClusteringGrid$ function is called to create an index structure based on a grid to map the trajectories' points (line 7). This structure is used to reduce the complexity of queries executed by the clustering algorithm [5], [19], [20]. In this work, a DBScan-based algorithm is used to cluster the trajectories' points in sequential order, the algorithm minPoint parameter was defined with value 1. When a current trajectory is being processed, all near trajectories are indexed in an auxiliary structure to be used in the grouping phase of the TODS.

B. Grouping

The grouping phase uses shape similarity to group trajectories. The Algorithm 2 details the $GetTrajectoriesGroups$ function (line 8) of Algorithm 1. The algorithm inputs are: the set of candidates C and structure index idx. Both variables are produced by recovering phase. The algorithm output is a set of groups (GT), where each group contains trajectories with a similar path to connect SR and ER.

Algorithm 2 TODS - Trajectory grouping

```
Require: $C, idx$
 1: $SortTrajectories(C)$
 2: $GT \leftarrow CreateSetStructure()$
 3: for all $c \in C$ do
 4:     $currentGroup \leftarrow empty$
 5:     for all $G \in GT$ do
 6:         if $FitInGroup(c, G, idx)$ then
 7:             $currentGroup \leftarrow G$
 8:             $break$
 9:         end if
10:     end for
11:     if $currentGroup = empty$ then
12:         $currentGroup \leftarrow CreateGroup()$
13:         $AddGroup(GT, currentGroup)$
14:     end if
15:     $AddCandidate(currentGroup, c)$
16: end for
17: return $GT$
```

The Algorithm 2 starts ordering the candidates list in ascending order by the number of points (line 1). For each candidate, the algorithm starts cleaning the variable $currentGroup$ (line 4). After that, it compares the current candidate c with the groups in GT set (line 6). If the candidate fits in an existent group then the variable $currentGroup$ receive that respective group and the search ends (lines 7-8). The $FitInGroup$ function uses the index structure to check the distance of all points belonged to c in the current trajectories cluster group G. If no group G has been found (line 11), then a new group is created to assign the candidate (line 12) and the new group is added to the GT set. Finally, the candidate c is allocated to the $currentGroup$ (line 15) and the next candidate is taken to the compared (line 3). This process guarantees that all candidates will be assigned to a group.

C. Separation

In this phase, the function $GetStandardTrajectories$ in line 9 of Algorithm 1 is executed to define the set of standard groups (STG) and the set of alternative groups ($NSTG$). This function performs the Definition 7 by ordering the trajectory groups GT in descending order and selecting the top KS groups with the highest number of trajectories as the set of standard groups (STG). The remaining groups are allocated in the set of alternative groups ($NSTG$).

D. Segmentation

The segmentation phase called by Algorithm 1 in line 10 is described in the Algorithm 3. The trajectory segmentation process is applied to the candidates of the alternative groups set ($NSTG$). The input parameters of Algorithm 3 are: set of groups (GT), set of standard groups (STG), set of alternative groups ($NSTG$), clustering distance (D) and angular difference threshold (θ). The algorithm returns the alternative routes segments separated in standard and alternative sets.

In Algorithm 3, the segmentation process is performed in two steps, the first one executes the segmentation by clustering the trajectories using distance, the second one executes the segmentation correction by assessing the angular difference among trajectories segments, as stated in Definition 8. The segmentation by distance (lines 3-5) identifies all trajectories points as standard or alternative using the clustering function $HasNear$. This function classifies a given point as standard due to their proximity to points from trajectories in STG. The trajectory segmentation correction (lines 6-11) reclassifies head and tail points from alternative segments by calculating the angular difference between them and the standard segments using $BackExtension$ and $FrontExtension$ functions, respectively. After all candidates points have been classified, the $Split$ function creates a route composed of standard and alternative segments, DS and NDS sets, respectively (lines 12-13). Finally, the route is included in the routes set (line 15) which is returned by the algorithm (line 16).

Algorithm 3 TODS - Trajectory segmentation

```
Require: $GT, STG, NSTG, D, \theta$
 1: $res \leftarrow CreateList()$
 2: for all $C \in NSTG$ do
 3:     for all $p \in C.points$ do
 4:         $p.std \leftarrow HasNear(STG, p, D)$
 5:     end for
 6:     for all $p \in C.points$ do
 7:         if $p.std = FALSE$ then
 8:             $BackExtension(p, C, STG, D, \theta)$
 9:             $FrontExtension(p, C, STG, D, \theta)$
10:         end if
11:     end for
12:     $R \leftarrow CreateRoute()$
13:     $Split(R.DS, R.NDS, C)$
14:     $AddRoute(res, R)$
15: end for
16: return $res$
```

V. Results

In order to evaluate the TODS algorithm, was performed a qualitative and a quantitative analysis. The qualitative analysis evaluates the segmentation process in order to identify if it is suitable for detecting standard routes changes during the day. This evaluation was carried out using the Google Maps API [6] to support the visualization of the results in the road network city map. Besides that, the quantitative analysis evaluates the algorithm runtime efficiency. The runtime analysis of each TODS phase has been taken and the TODS index structure efficiency was compared with the well-known tree indexes RNNSearch [21] and CoverTree [23] methods. Both algorithms are implemented in the Java Framework Smile [21].

Finally, a comparison with the TRA-SOD (Trajectory Outlier Detection Algorithm) [4] was performed to assess the proposed method (TODS) with a similar algorithm. The TRA-SOD algorithm is a clustering algorithm that takes into consideration space and time context to find outliers in moving objects trajectories. The main difference between TODS and TRA-SOD is that the former has the capability to segment trajectories in standard and alternative segments. Besides that, TODS allow to configure the number of standard paths, different from TRA-SOD that can classify multiple paths as standard. This happens in TRA-SOD due to the difficulty to define the appropriated value for the k-nearest neighborhood parameter. The k-nearest neighborhood parameter is not trivial to estimate because it is necessary to understand quite well the dataset to quantify a good value to execute the clustering.

The whole analysis was performed using two heterogeneous datasets. The first dataset contains data collected by smartphone users in Joinville - Brazil. The second one dataset was collected by taxi drivers in San Francisco - USA. Both cities were selected due to the availability of data and the complex road networks. The Brazil dataset contains approximately 5000 trajectories collected during 2013 by 10 people using 8 different smartphone devices. In contrast, the USA dataset contains approximately 25000 trajectories collected by 500 taxi drivers during 30 days from May to June 2008. Table I presents the datasets summary where it can be observed that trajectories collected in Brazil are denser than the ones collected in the USA. This characteristic impacts directly on the algorithm performance.

In order to perform the experiments, the values defined to

TABLE I
DATASETS STATISTICAL SUMMARY ABOUT COLLECTED TRAJECTORIES AND POINTS (COORDINATES).

Statistics	Brazil	EUA
Trajectories quantity	5083	24999
Trajectory points quantity (Mean)	252.9	45.1
Trajectory points quantity (SD)	494.9	32.9
Trajectory points quantity (Max)	4810	1127
Trajectory points quantity (Min)	1	1
Points quantity	1280484	1103663
Adjacent points distance (Mean)	6.4(m)	48.9(m)
Adjacent points distance (SD)	7.9(m)	908.1(m)
Adjacent points distance (Max)	234.9(m)	669323.1(m)
Adjacent points distance (Min)	0.0(m)	0.0(m)

TRA-SOD parameter are the ones suggested in [4], which are: $maxDist = 50m$ and $minSup = 4$. While, the parameter values defined in the algorithm TODS are: $SH = 0h$, $EH = 23h$, $I = 35m$, $SD = 100m$, $\sigma = 99.7\%$, $D = 50m$, $\theta = 30$ and $KS = 1$. Although, the TODS algorithm has more parameters compared to TRA-SOD, some parameters like SD, σ and I can be automatically defined in future studies. These parameter values were empirically defined based on analysing performed using the selected datasets as well as by the literature related suggestions.

Regarding the development of the algorithms, they were developed using Java (JDK version 1.8.0_121 [10]) and the datasets were stored using MongoDB version 3.2.12 [17]. MongoDB was chosen due to its geospatial handling capability. Concerning the hardware specification, a computer with Intel Core I5 processor and 6GB RAM was used to execute the analysis.

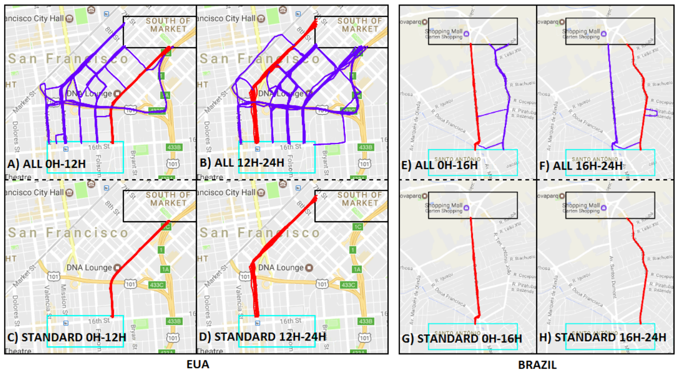


Fig. 4. Trajectories temporal evolution

A. Temporal evolution

As previously described, the TODS algorithm uses the time context to filter trajectories to be analyzed. Figure 4 demonstrates how time evolution modifies the usage of the roads in different periods of the day. Images A, B, C, and D show the scenario 1 with trajectories from the United States and images E, F, G, and H depict the scenario 2 with trajectories from Brazil. Images A, B, E, and F show the classification of all trajectories (standard and alternatives) while C, D, G, and H show only standard trajectories. These trajectories were filtered at two different time intervals. For the United States the values were 0H-12H and 12H-24H and for Brazil, the values were 0H-16H and 16H-24H. As can be seen in Figure 4, the standard route is sensitive to the time of the day, as observed in images C, D, G, and H, where the standard route of both scenarios changed during the day. In scenario 1, it has been selected 72 trajectories from 0H to 12H (image A), they were classified into 21 groups and the standard route has 18 trajectories (image C). In contrast, also in scenario 1, it was selected 38 trajectories from 12H to 24H (image B), they were classified in 15 groups and the standard route has 14 trajectories (image D). Besides that, in scenario 2, it has been selected 36 trajectories from 0H to 16H (image E), they were classified in 5 groups and the standard route has 29 trajectories (image G). However, also in scenario 2, it has been selected 26 trajectories from 16H to 24H (image F), they were classified in 5 groups and the standard route has 15 trajectories (image H).

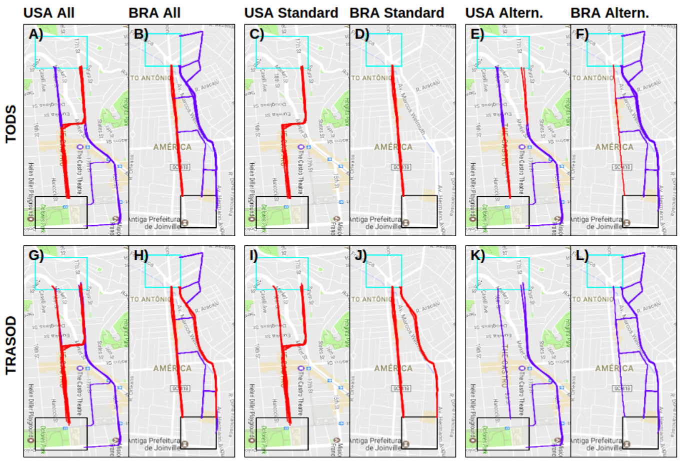


Fig. 5. Comparison between TODS and TRA-SOD

B. Comparison between TODS and TRA-SOD

TODS and TRA-SOD algorithms were compared using Brazil and USA datasets. To execute the tests, temporal outliers were not considered as in [4], only the spatial classification is used due to different methods applied to deal with time contexts in both algorithms. The algorithms were executed using the same regions. One of the obtained results can be seen in Figure 5, where images A, B, G, and H show the complete classification, images C, D, I and J show the trajectories belonged to standard routes, and the images E, F, K and L show the trajectories belonged to alternative routes. Regarding the results, for the tests executed with the USA dataset, TRA-SOD classified 12 trajectories in 2 standard routes (image I) and 4 trajectories in 3 alternative routes (image K), while the TODS classified 8 trajectories in 1 standard route (image C) and 8 trajectories in 3 alternative routes, segmented into 2 standards and 3 alternative segments (image E). In contrast, the tests executed in the Brazil dataset, TRA-SOD algorithm found 64 trajectories in 2 standard routes (image J) and 7 trajectories in 7 alternative routes (image L), while the TODS algorithm found 45 trajectories in 1 standard route (image D) and 26 trajectories in 8 alternative routes, segmented into 2 standards and 8 alternative segments (image F).

The alternative trajectories results using the Brazil dataset (image F) were expanded for visualization purposes in Figure 6. This figure demonstrates all alternatives groups and their respective segments. In this figure, the group 0 has 19 trajectories, while the remaining have only one.

Analyzing the trajectories classification it is possible to observe that TRA-SOD found more standard routes than TODS. It is explained due to the fact that TRA-SOD algorithm is not sensitive to the amount of data, but rather to a fixed value of K neighbors used in clustering. Besides that, when data collection grows, TRA-SOD has it classification capacity decreased. It happens because when there are many similar trajectories the algorithm starts to classify a greater number of trajectories in the standard routes. In addition, the lack of segmentation technique in TRA-SOD difficulties the analysis to find the moments that a given trajectory is following a standard route. In contrast, the TODS algorithm by means of the segmentation processes turns easy the process of identifying when a trajectory segment is standard or alternative.

C. Algorithm runtime performance

In order to evaluate the TODS runtime performance, 7000 samples (3500 from Brazil and 3500 from the USA) were selected from randomly generated pairs of regions that have, at least, one alternative route connecting them. Figure 7 A shows the sample distribution regarding number of trajectories per number of points before TODS pre-processing phase. Figure 7 B uses a boxplot to shows the runtime performance of TODS concerning the number of trajectory points. The number of point has been chosen as assessment criteria because it is the most relevant trajectories' characteristic that impacts the algorithm runtime performance. As observed in Figure 7 B, the runtime grows as long as the trajectory number of points increases. Some outliers runtime performances, represented as circle in Figure 7 B, were detected. This was probably caused by the Java Garbage Collector mechanism that executes extra operations during the trajectories processing. The more points the trajectory has the more impact Java Garbage Collector causes. Besides that, it is observed that trajectories with 0 and 30k points have a very similar execution time in the second and third quartiles. This indicates that the algorithm is stable and hence spends almost the same time to provide the result, according to the number of points. However, for trajectories with 30k to 70k points the runtime is more unstable due to the reduced number of analyzed trajectories to support better runtime execution analysis.

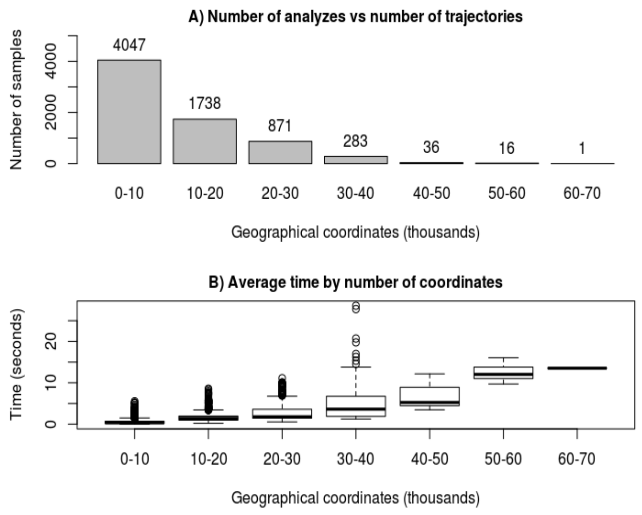


Fig. 7. Samples distribution and runtime

When analyzing the runtime performance of TODS algorithm considering each one of its phases, in Figure 8 is possible to see that in average more than 90% of execution time is spent in step 1 (Recovery). This is explained because the process of generating the clusters and creating the index structure is very time consuming. On the other way around, the remaining phases have reduced execution time due to the benefits brought by the index structure that speed up the search for neighbor trajectories.

As the TODS creation index structure is one of the most expensive process, it has been evaluated against two of the

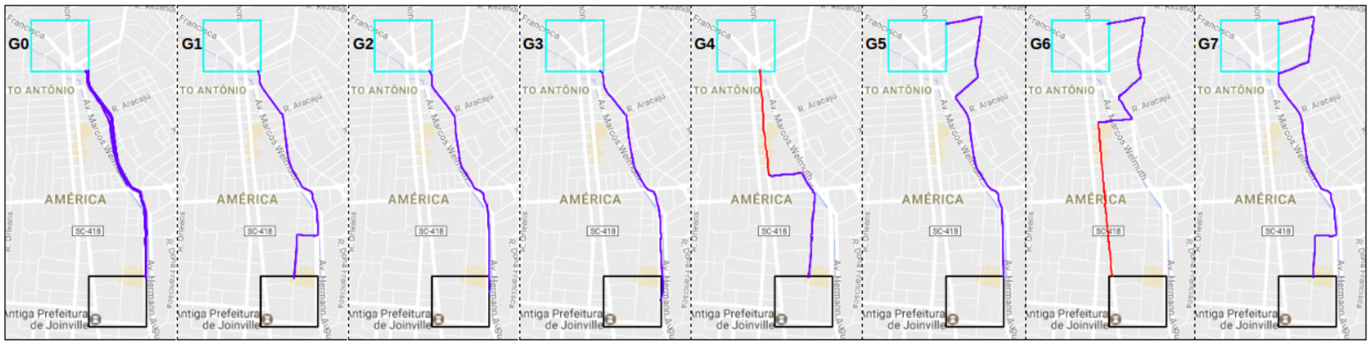


Fig. 6. Alternative groups classified by TODS

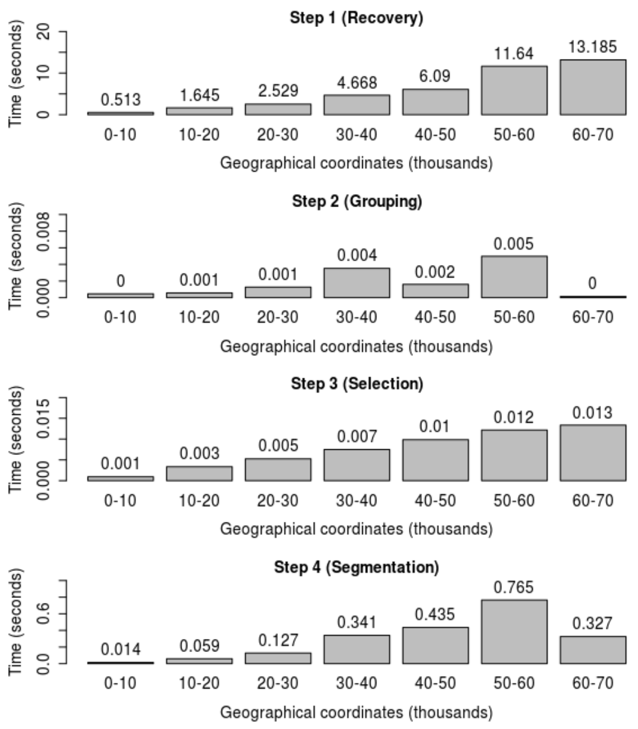
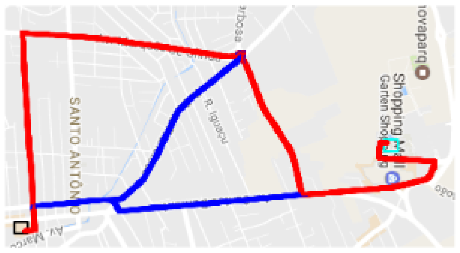


Fig. 8. Runtime by steps

most well-know indexing algorithms available in the literature, that are: CoverTree [23] and RNNSearch [21]. To perform this assessment, the analyzed index algorithms were implemented in Java and were tested under a clusterization process handled by a standard implementation of DBScan algorithm [24]. The test scenarios were generated based on routes suggested by Google Maps [6] between two predefined regions of Joinville. One example of those scenarios is presented in Figure 9 (A). These routes were used as templates to artificially generate a suitable amount of similar trajectories to compare the index algorithms. The generation of artificial trajectories consists of interpolating points with a random applied noise in latitude and longitude of those points belonged to the routes suggested by Google Maps using $50m$ as default seed. To execute the tests, five scenarios were generated with 100, 200, 300, 400, and 500 trajectories resulting in 15000, 30000, 45000, 60000, 75000 points, respectively.

Concerning the results, Figure 9 (B) presents the clustering runtime performance for each one of the analyzed indexing algorithms. As can be seen, TODS indexing algorithm performed better than CoverTree and RNNSearch algorithms. This is explained because TODS indexing algorithm organizes the trajectories' points in a $n \times m$ grid structure, where n and m are defined considering that each grid cell presents a square of $100m^2$ and the grid must represent the entire search space of trajectories. Besides that, TODS has some performance improvements tailored to deal with issues presented in the TODS approach. For example, TODS index structure does not maintain in memory points already processed. Finally, CoverTree does not show suitable performance for trajectories with high number of points because it was conceived to be efficient with small amount of data [23].



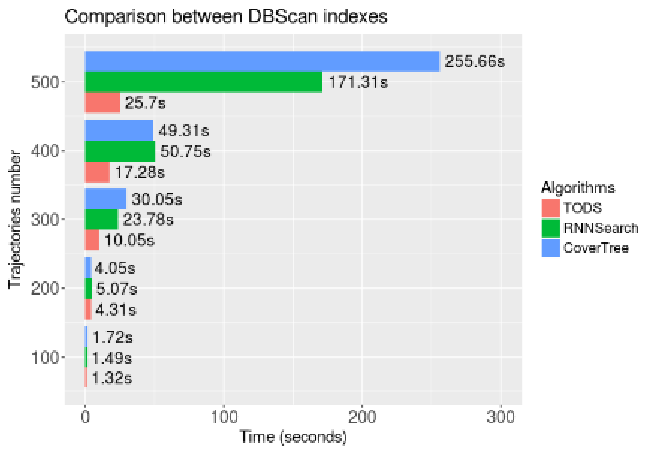


Fig. 9. Clustering algorithms runtime analysis

VI. Conclusion and Future Works

This work proposes an algorithm to suggest alternative routes for ordinary drivers named TODS. This algorithm uses a clustering technique to group and segment car trajectories as standard and alternative segments by analyzing roads' usage frequency. The less used segments are identified as outliers and are suggested as alternative driving directions for ordinary drivers to allow them to avoid traffic jams.

The algorithm was tested in two trajectory datasets with different point collection distribution. Based on the performed analysis, it has been observed that TODS is capable to support different real-world scenarios, giving to users different alternative routes considering the road usage during the time of the day. A comparison with TRA-SOD algorithm was performed and shows that TODS is more effective in presenting alternative routes for ordinary drivers due to trajectories segmentation in standard and alternative parts. Besides that, TODS also better suggests the most relevant standard routes by approaching the road usage as a trajectories grouping by distance. This avoids the wrong classification of trajectories as standard. Finally, the last analysis showed that standard routes usage is influenced by the time of the day as both standard and alternative routes changes along the day. One of the TODS limitations is the configuration of the angle parameter used to perform the segmentation, because it is necessary to adapt it to different road network scenarios. For example, in highway roads the angle must be small as the roads diverge smoothly and hence it should be earlier identified. On the other way around, in downtowns the angle must be big to avoid unexpected segmentation.

As future works, first, it is necessary to evaluate self-parametrization methods to reduce the empirical parameters configuration. Another proposed study is to use statistical methods to analyze bottlenecks in the city and suggest improvements in the road network. Finally, a method to identify experts users could be developed to improve the alternative routes suggestion.

References

[1] A. Anagnostopoulos, M. Vlachos, M. Hadjieleftheriou, E. Keogh and P. Yu, ”Global distance-based segmentation of trajectories”, Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’06, 2006.

[2] A. Bessa, F. Silva, R. Nogueira, E. Bertini and J. Freire, ”RioBusData: Outlier Detection in Bus Routes of Rio de Janeiro”, CoRR, 2016.

[3] M. Buchin, A. Driemel, M. Kreveld and V. Sacristán, ”Segmentation trajectories: A framework and algorithms using spatiotemporal criteria”, JOSIS, , pp. 33-63, 2011.

[4] V. Fontes and V. Bogorny, ”Discovering Semantic Spatial and Spatio-Temporal Outliers from Moving Object Trajectories”, CoRR, 2013.

[5] E. Knorr, R. Ng and V. Tucakov, ”Distance-based outliers: algorithms and applications”, The VLDB Journal The International Journal on Very Large Data Bases, vol. 8, no. 3-4, pp. 237-253, 2000.

[6] Google.com, 2018. [Online]. Available: https://www.google.com/maps. [Accessed: 18- Mar- 2018].

[7] J. Lee, J. Han and X. Li, ”Trajectory Outlier Detection: A Partition-and-Detect Framework”, 2008 IEEE 24th International Conference on Data Engineering, 2008.

[8] H. Liu and M. Schneider, ”Similarity measurement of moving object trajectories”, Proceedings of the Third ACM SIGSPATIAL International Workshop on GeoStreaming - IWGS ’12, 2012.

[9] W. Luo, H. Tan, L. Chen and L. Ni, ”Finding time period-based most frequent path in big trajectory data”, Proceedings of the 2013 international conference on Management of data - SIGMOD ’13, 2013.

[10] ”java.com: Java + You”, Java.com, 2018. [Online]. Available: https://www.java.com/en/. [Accessed: 18-Mar-2018].

[11] M. Ma, H. Ngan and W. Liu, ”Density-based Outlier Detection by Local Outlier Factor on Largescale Traffic Data”, Electronic Imaging, vol. 2016, no. 14, pp. 1-4, 2016.

[12] C. Panagiotakis, N. Pelekis and I. Kopanakis, ”Trajectory Voting and Classification Based on Spatiotemporal Similarity in Moving Object Databases”, Advances in Intelligent Data Analysis VIII, pp. 131-142, 2009.

[13] C. Panagiotakis, N. Pelekis, I. Kopanakis, E. Ramasso and Y. Theodoridis, ”Segmentation and Sampling of Moving Object Trajectories Based on Representativeness”, IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 7, pp. 1328-1343, 2012.

[14] S. Sankararaman, P. Agarwal, T. Mølhave, J. Pan and A. Boedihardjo, ”Model-driven matching and segmentation of trajectories”, Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL’13, 2013.

[15] Z. Zhou, W. Dou, G. Jia, C. Hu, X. Xu, X. Wu and J. Pan, ”A method for real-time trajectory monitoring to improve taxi service using GPS big data”, Information & Management, vol. 53, no. 8, pp. 964-977, 2016.

[16] J. Zhu, W. Jiang, A. Liu, G. Liu and L. Zhao, ”Effective and efficient trajectory outlier detection based on time-dependent popular route”, World Wide Web, vol. 20, no. 1, pp. 111-134, 2016.

[17] ”MongoDB for GIANT Ideas”, MongoDB, 2018. [Online]. Available: https://www.mongodb.com/. [Accessed: 18-Mar-2018].

[18] Y. Mao, H. Zhong, X. Xiao and X. Li, ”A Segment-Based Trajectory Similarity Measure in the Urban Transportation Systems”, Sensors, vol. 17, no. 12, p. 524, 2017.

[19] ”A faster algorithm for DBSCAN”, Master, Technische Universiteit Eindhoven, 2013.

[20] J. Gan and Y. Tao, ”DBSCAN Revisited”, Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD ’15, 2015.

[21] ”Smile - Statistical Machine Intelligence and Learning Engine”, Haifengl.github.io, 2018. [Online]. Available: http://haifengl.github.io/smile/clustering.html#dbscan. [Accessed: 18-Mar-2018].

[22] ”Double-Digit Growth Forecast for the Worldwide Big Data and Business Analytics Market Through 2020 Led by Banking and Manufacturing Investments, According to IDC”, IDC: The premier global market intelligence company, 2018. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS41826116. [Accessed: 18-Mar-2018].

[23] A. Beygelzimer, S. Kakade and J. Langford, ”Cover trees for nearest neighbor”, Proceedings of the 23rd international conference on Machine learning - ICML ’06, 2006.

[24] M. Ester, H. Kriegel, J. Sander and X. Xu, ”A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise.”, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining - KDD ’96, pp. 226-231, 1996.

[25] Jing Yuan, Yu Zheng, Xing Xie and Guangzhong Sun, ”T-Drive: Enhancing Driving Directions with Taxi Drivers’ Intelligence”, IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 1, pp. 220-232, 2013.